Der TRADE–Trader: Ein Basisdienst offener verteilter Systeme

K. Müller, M. Merz, W. Lamersdorf

Universität Hamburg
Fachbereich Informatik — Datenbanken und Informationssysteme
Vogt–Kölln–Str. 30 — D–22527 Hamburg

[kmueller|merz|lamersd] @ dbis1.informatik.uni-hamburg.de

Zusammenfassung

Dem Trader als wichtiger Basisdienst offener verteilter Systeme kommt zunehmend eine zentrale Bedeutung zu. Angesichts der Komplexität verteilter Systeme und der Vielzahl und Vielfältigkeit der angebotenen Dienste bietet der Trader adäquate Konzepte zur Strukturierung der Dienste an und stellt Mechanismen für die Dienstvermittlung und –verwaltung bereit. Dieser Artikel stellt die Funktionalität und Implementierung des TRADE–Traders vor, welcher im Rahmen des COSM/TRADE–Projektes[1] an der Universität Hamburg prototypisch auf Basis des Distributed Computing Environment (DCE) realisiert ist. Am Beispiel der Implementierung des TRADE–Traders wird im vorliegenden Beitrag gezeigt, wie eine modulare Erweiterung bestehender verteilter Systemarchitekturen um Funktionen des Tradings und eine weitgehende Integration mit den bereits durch die Systemarchitektur bereitgestellten Diensten, insbesondere den vorhandenen Dienstvermittlungs– und Verwaltungsdiensten, erreicht werden kann.

1 Einleitung

Mit der stetigen Zunahme der Komplexität heutiger verteilter Systeme ergibt sich die Notwendigkeit nach Strukturierungsmöglichkeiten, die eine effektive und effiziente Nutzung der Dienste eines verteilten Systems ermöglichen. Eine zentrale Problematik innerhalb von verteilten Systemen ist dabei die Verwaltung und der Zugriff auf *Dienste* bzw. Ressourcen, die systembedingt global verteilt sind. Die Aufgabe der Verwaltung von Diensten wurde bisher bzw. wird durch Namensdienste erbracht, die den Systemnutzern (Anwender, Anwendungsprogramme) einfache Möglichkeiten zur Ablage, zum Suchen und Auffinden von Diensten bereitstellen. Neben der ursprünglichen Funktionalität der einfachen Abbildung von logischen Diensten (Namen) auf physikalische Netzadressen unterstützen Namensdienste zum Teil auch attributbasierte Anfragen, so daß eine Auswahl von Diensten über Eigenschaften der registrierten Dienste durchgeführt werden kann. Ein bekanntes Beispiel für einen derartigen, attributbasierten Namensdienst ist der globale Namensdienst X.500 [Neu89].

In Hinblick auf heutige verteilte Systeme mit ihrer Vielzahl und Vielfalt an angebotenen Diensten (auch als *Dienstemarkt* bezeichnet [ML93]) reichen derartige, relativ einfache Mechanismen jedoch nicht aus, um diese Dienste adäquat verwalten und vermitteln zu können und somit für mögliche Dienstnutzer effizient nutzbar zu machen. Vor dem Hintergrund dieser Einschränkungen hat sich deshalb in den letzten Jahren eine relativ junge Forschungsrichtung etabliert, die eine Fortführung und Weiterentwicklung der Grundprinzipien der herkömmlichen Namensverwaltung anstrebt. Der Kern der Untersuchungen manifestiert

[1] **C**ommon **O**pen **S**ervice **M**arket / Service **Tra**ding and Coordination **E**nvironment

sich hierbei an der Funktion des sog. *Traders* oder *Brokers*, welcher zur Zeit im Rahmen der internationalen Standardisierung *Open Distributed Processing (ODP)* als sog. *ODP Trading Function* [ODP92] definiert wird.

Auf dem Hintergrund dieser Standardisierungsbemühungen ist es deshalb notwendig, die sinnvolle Integration der Trading–Funktion in bestehende und sich neu entwickelnde verteilte Systemarchitekturen zu untersuchen. Dieses betrifft vor allem die Einbeziehung vorhandener Dienste der jeweiligen Systemarchitekturen, insbesondere der Dienstvermittlungs– und Dienstverwaltungsfunktionen, um eine kohärente Integration der Trading–Funktion zu gewährleisten. Der Trader ist hierbei zunächst als *Add-on*–Komponente zu verstehen, mit der Möglichkeit, zu einem späteren Zeitpunkt selbst ein integraler Bestandteil der Systemarchitektur zu werden. Im folgenden betrachtet dieser Artikel die Integration der Trading–Funktion in das *Distributed Computing Environment (DCE)* der Open Software Foundation [Fou92], welches zunehmend an Bedeutung und Verbreitung in der Entwicklung verteilter Anwendungen gewinnt. Die Realisierung der Trading–Funktion erfolgt hierbei durch den TRADE–Trader, welcher im Rahmen des COSM/TRADE–Projektes [MJM94, MML94] an der Universität Hamburg prototypisch implementiert ist und als Basis für laufende Forschungsarbeiten im Bereich Dienstvermittlung und –verwaltung dient.

Der Artikel gliedert sich im weiteren wie folgt: Der folgende Abschnitt beschreibt die grundlegende Aufgabe eines Trading–Dienstes und zeigt zwei wichtige Strukturierungskonzepte zur Vermittlung und Verwaltung von Diensten auf. Anschließend wird eine Analyse der DCE–Systemarchitektur in Hinblick auf vorhandene Systemdienste zur Dienstvermittlung und –verwaltung vorgenommen. Insbesondere werden der DCE–Namensdienst und die für den Dienstzugriff notwendigen Bindungsmechanismen untersucht. Auf Basis dieser Analyse wird der TRADE–Trader und seine Integration in die DCE–Systemarchitektur vorgestellt sowie seine Implementierungstechniken beschrieben. Abschließend erfolgt eine Zusammenfassung und ein Ausblick auf weitere Forschungs– und Entwicklungsvorhaben.

2 Die Trading–Funktion

Die zentrale Aufgabe eines Traders ist die Vermittlung und Verwaltung von Diensten in offenen verteilten Systemen. Der Trader bietet hierzu Mechanismen an, die eine Strukturierung der verschiedenartigen Dienste ermöglichen und den Dienstnehmer beim Auffinden geeigneter Diensterbringer unterstützen. Die Funktionalität eines Traders läßt sich am besten mit der eines „Gelbe Seiten“–Dienstes vergleichen, der Dienste kategorisiert und eine Suche über Eigenschaften der Dienste, sog. *Dienstattribute*, anbietet. Das formale Konzept für die Kategorisierung von Diensten durch den Trader ist der *Diensttyp* [ODP92]. Der Diensttyp setzt sich zusammen aus einem *Schnittstellentyp*, welcher die operationale Schnittstelle eines Dienstes anhand von Operationstypen spezifiziert, und den *Dienstattributtypen*, die die Eigenschaften eines Dienstes näher spezifizieren. Ein weiteres Strukturierungsmittel des Traders sind die *Kontexte*. Diese ermöglichen die strukturierte Ablage von Dienstangeboten, z. B. in Form einer hierarchisch organisierten Kontextstruktur.

Die Interaktionen der zu verwaltenden Diensterbringer und der Dienstnehmer mit dem Trader verdeutlicht Abbildung 1. Ein Diensterbringer, der sog. *Exporteur*, registriert sein Dienstangebot beim Trader (Schritt 1). Hierzu gibt er u. a. den Diensttyp, die aktuellen Werte der Dienstattribute und den Kontext an, in den das Dienstangebot exportiert werden soll. Ein Dienstnehmer, der sog. *Importeur*, richtet seine Suchanfrage nach einem

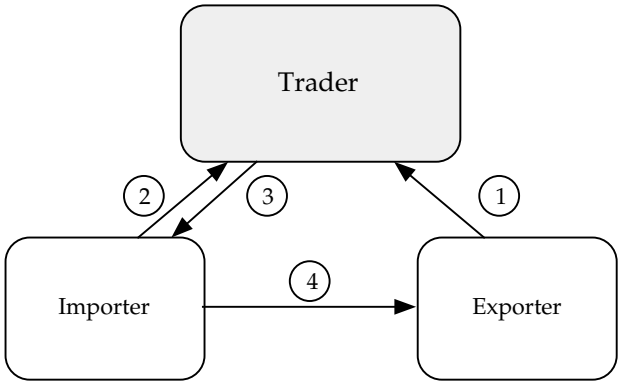

Abbildung 1: Der Trader und seine Nutzer

bestimmten Diensterbringer an den Trader (Schritt 2). Die Suchanfrage setzt sich dabei unter anderem aus dem gewünschten Diensttyp, den gewünschten Diensteigenschaften sowie einem Suchkontext zusammen. Aufgrund dieser Suchkriterien ermittelt der Trader passende Dienstangebote, wählt gegebenfalls das am besten geeignete Dienstangebot aus und übermittelt dem Dienstnehmer die für die anschließende Interaktion mit dem Diensterbringer notwendigen Bindungsinformationen (Schritt 3). Mit diesen Informationen kann der Dienstnehmer dann direkt die Operationen des Diensterbringers ausführen (Schritt 4).

3 DCE–Mechanismen zur Vermittlung und Verwaltung von Diensten

Das Distributed Computing Environment (DCE) der Open Software Foundation stellt eine Plattform zur Entwicklung verteilter Anwendungen bereit, die auf dem Client/Server–Modell basieren. DCE versteht sich hierbei als sog. *Middleware* [Ber93], mit deren Hilfe verteilte Anwendungen auf verschiedenen heterogenen Systemen plattformunabhängig entwickelt werden können. Hierzu bietet DCE eine Menge von miteinander integrierten Basisdiensten an, die die Programmierung verteilter Anwendungen unterstützen. Grundlegende DCE–Dienste sind der *Remote Procedure Call (DCE RPC)*, der *Thread Service*, der *Cell Directory Service (CDS)*, der *X.500–konforme Global Directory Service (GDS)*, der *Security Service* und der *Distributed Time Service (DTS)*. Weitere Dienste sind der *Distributed File Service (DFS)* und der *Diskless Client Support*. Für jeden DCE–Basisdienst wird dem Anwendungsentwicker eine entsprechende Programmierschnittstelle angeboten, wobei der Cell Directory Service über zwei verschiedene Schnittstellen verfügt.

3.1 DCE–Dienstverwaltung und –zugriff

Die grundlegende Verwaltungs– und Strukturierungseinheit von DCE ist die sog. *Zelle*. Jede Zelle verfügt über einen eigenen Security Service, einen eigenen Cell Directory Service sowie einen eigenen Distributed Time Service. Die Ablage von Informationen über sämtliche Dienste innerhalb der Zelle einschließlich der DCE–Basisdienste erfolgt im CDS. Der Zugriff auf Informationen anderer Zellen ist durch die Nutzung des Global Directory Service möglich, der einen globalen Namensraum zur Verbindung verschiedener DCE–Zellen zur Verfügung

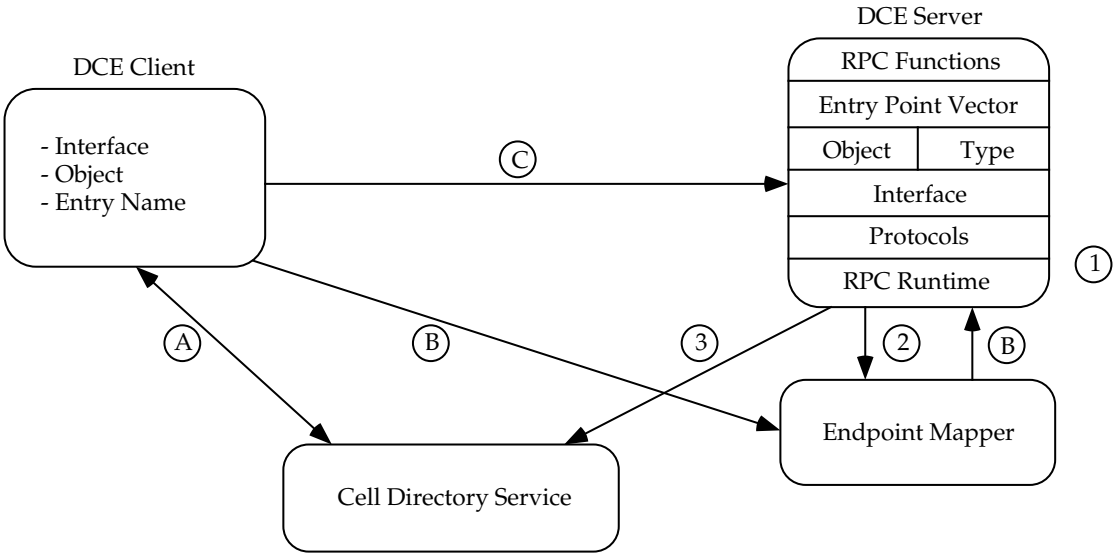

Abbildung 2: Bindung von Dienstnehmer und Diensterbringer mittels CDS

stellt[2].

Der zentrale Begriff der DCE–Dienstverwaltung ist die *Schnittstelle (interface)*. Damit ein Diensterbringer eine Schnittstelle anbieten kann, muß diese vorher in einer abstrakten *Schnittstellendefinitionssprache (interface definition language, IDL)* beschrieben sein. Diese beinhaltet Informationen über die angebotenen Operationstypen und die dazugehörigen Datentypen für die Parameter– und Ergebniswerte. Zur eindeutigen Identifikation von Schnittstellen dienen sog. *UUIDs (universal unique identifier)*, die Bestandteil der Beschreibung sind. Diese entsprechen den im Trader definierten *Schnittstellentypen*. Zusätzlich beinhaltet die Beschreibung eine *Versionsnummer*, die eine eingeschränkte Möglichkeit zur Kompatibilität von Schnittstellen mit gleicher UUID ermöglicht. Kompatibilität beschränkt sich hierbei auf das Hinzufügen neuer Operationen zu einer Schnittstelle, womit sich eine Form des Polymorphismus, der Inklusionspolymorphismus [CW85], erreichen läßt. Für eine korrekte Definition der Kompatibilitätsbeziehungen zwischen Schnittstellentypen ist der Anwendungsentwickler verantwortlich. Im folgenden soll nun die Bindung eines Dienstnehmers zu einem bestimmten Diensterbringer mit der gewünschten Schnittstelle unter Nutzung des Cell Directory Service beschrieben werden. Abbildung 2 zeigt die beteiligten Systemkomponenten und kennzeichnet die im folgenden genauer beschriebenen Ablaufschritte. Zum Anbieten einer Schnittstelle sind auf Diensterbringerseite die folgenden Schritte notwendig:

1. Als erstes wird dem RPC–Laufzeitsystem der angebotene Schnittstellentyp (in Form eines sog. *Interface Handle*) bekannt gegeben. Zusätzlich werden mit jeder Schnittstelle eine lokale Typkennung (*type UUID*), eine Objektkennung (*object UUID*) zum global eindeutigen Zugriff auf die angebotene Schnittstelle und ein *Entry Point Vector* verwaltet, der als lokaler Verweis auf die jeweiligen Schnittstellenprozeduren dient. Ein Diensterbringer kann auch mehrere Schnittstellen verschiedenen Typs gleichzeitig anbieten.

2. Anschließend werden sog. *Bindungsvektoren* generiert, welche Informationen über die unterstützten Kommunikationsprotokolle und die dynamisch erzeugten Kommunika-

[2]Eine Alternative ist der Internet *Domain Name Service (DNS)*.

tionsendpunkte (*endpoints*) beinhalten. Diese werden zusammen mit dem Schnittstellentyp und der Objektkennung dem lokalen *Endpoint Mapper* übergeben. Dieser verwaltet die auf dem lokalen Knoten laufenden Diensterbringer und die Abbildung der Schnittstellentypen bzw. Objektkennungen auf die entsprechenden Kommunikationsendpunkte.

3. Als letzter Schritt werden die gleichen Informationen, bis auf die Kommunikationsendpunkte, unter einem bestimmten Namenseintrag im Cell Directory Service eingetragen.

Bevor nun Operationen des Diensterbringers aufgerufen werden können, muß der Dienstnehmer die folgenden Schritte ausführen:

A. Mit Aufrufen an den Cell Directory Service werden die unter dem gewünschten Namenseintrag abgelegten Bindungsinformationen erfragt. Hierbei müssen zusätzlich zum Eintragsnamen der Schnittstellentyp und die Objektkennung mit angegeben werden.

B. Mit den vom CDS erhaltenen Bindungsinformationen wird dann der erste entfernte Prozeduraufruf durchgeführt. Da jedoch in den Bindungsinformationen zunächst keine Kommunikationsendpunkte enthalten sind, wird der Aufruf automatisch durch das RPC–Laufzeitsystem zuerst an den auf dem Rechner des Diensterbringers lokalisierten Endpoint Mapper geleitet, der daraufhin die Bindungsinformationen um die entsprechenden Endpunkte erweitert und den Aufruf an den Diensterbringer weitergibt[3].

C. In folgenden entfernten Prozeduraufrufen wird dann mit diesen erweiterten, vollständigen Bindungsinformationen direkt auf den Diensterbringer zugegriffen.

3.2 DCE–Dienstvermittlung

DCE bietet selbst keinen Dienst zur Vermittlung von Diensterbringern an. Somit findet die Dienstauswahl ausschließlich durch den Dienstnehmer statt. Eine einfache Unterstützung erhält der Dienstnehmer durch die Verwaltungsfunktionen des Cell Directory Service. Im Vergleich zu den Unterstützungsmechanismen eines Traders sind die Möglichkeiten des CDS jedoch sehr eingeschränkt. Primär dient das CDS als einfacher, hierarchisch organisierter Verwaltungsdienst, welcher eine Ablage von Bindungsinformationen der unter einem bestimmten CDS–Namenseintrag registrierten Diensterbringer bereitstellt. Die von DCE speziell für die RPC–Programmierung bereitgestellte *NSI–Programmierschnittstelle* zur Interaktion mit dem CDS erlaubt jedoch nur namensbasierte Anfragen. Praktisch bedeutet dies, daß ausschließlich Informationen über CDS–Einträge mit bereits a–priori bekannten Namen durch den Dienstnehmer ermittelt werden können. Dies gilt auch für Gruppeneinträge und sog. *Profiles*, die eine zusätzliche Strukturierung der Diensterbringer ermöglichen. Da die NSI–Schnittstelle keine attributbasierte Suchoperation zum Finden geeigneter Diensterbringers bereitstellt, muß der Dienstnehmer selbst sämtliche CDS–Einträge sukzessive durchsuchen und auswerten. Die erlaubten Abfragekriterien beschränken sich hierbei ausschließlich auf den gewünschten Schnittstellentyp und die Objektkennung. Wie schon oben dargestellt werden dem Dienstnehmer auch Bindungsinformationen von Diensterbringern mit

[3]Alternativ kann sich der Dienstnehmer auch selbst direkt an den Endpoint Mapper wenden und die Bindungsinformationen ergänzen.

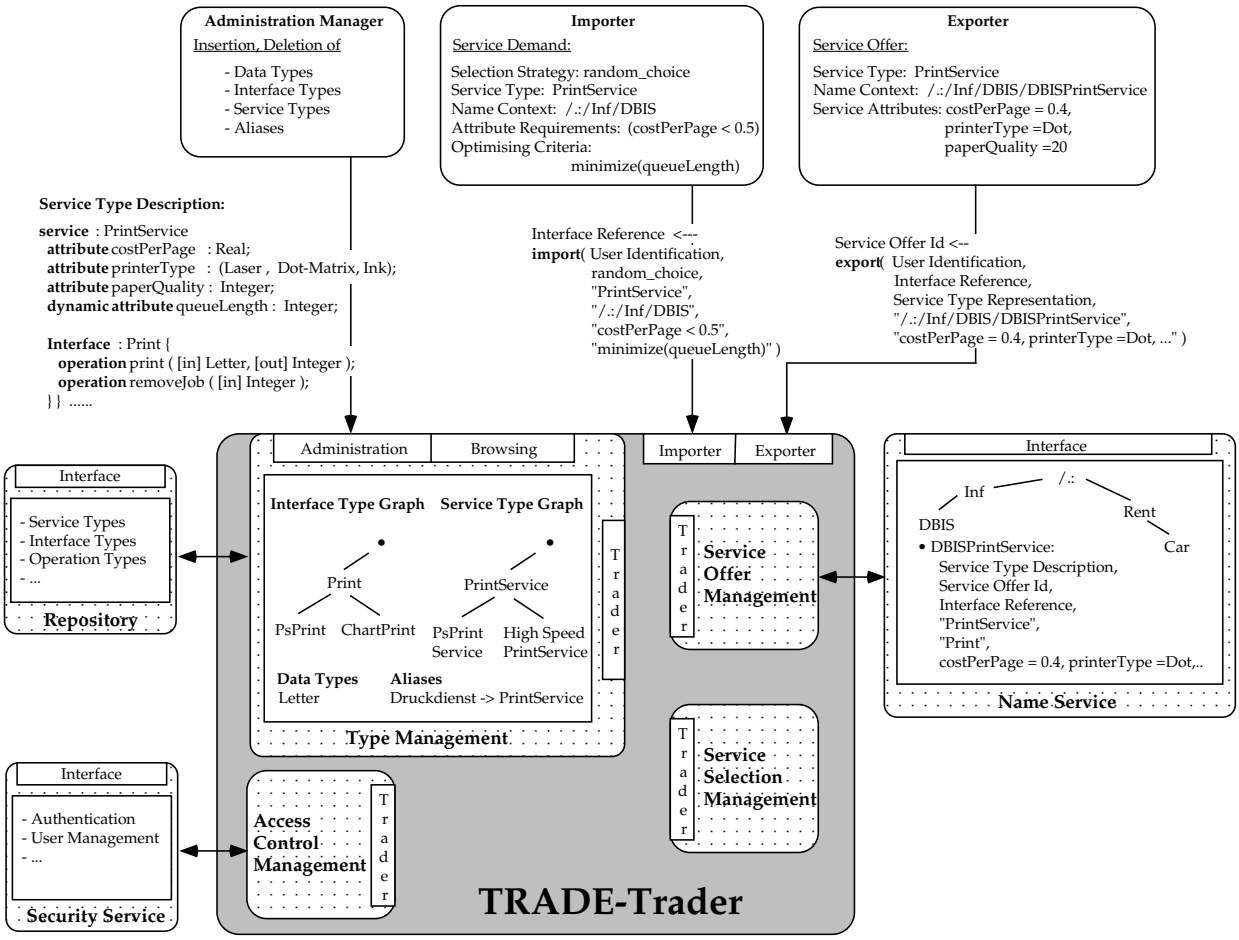

Abbildung 3: Die Architektur des TRADE–Traders

kompatiblen Schnittstellentypen zurückgegeben. Im Vergleich zum Trader als typverwaltende und typüberwachende Systemkomponente bietet DCE jedoch keine Mechanismen zur Überprüfung der Korrektheit der vom Programmierer definierten Kompatibilitätsbeziehungen zwischen Schnittstellentypen an.

4 Dienstvermittlung und –verwaltung innerhalb von DCE: Der TRADE–Trader

Im folgenden wird nun die Integration der Trading–Funktion in DCE anhand des im Rahmen des COSM/TRADE–Projektes an der Universität Hamburg prototypisch realisierten *TRADE–Traders* dargestellt. Als Implementierungsplattform dient eine UNIX–Rechnerumgebung mit mehreren vernetzten IBM RS/6000 Arbeitsplatzrechnern, die innerhalb einer DCE–Zelle verwaltet werden. Die Integration des TRADE–Traders in DCE erfolgt als zusätzlicher DCE–Dienst, wobei soweit wie möglich die in DCE vorhandenen Basisdienste, insbesondere der DCE RPC, die beiden Namensdienste CDS und GDS, der Security Service sowie implizit der Thread Service und der Distributed Time Service genutzt werden. Abbildung 3 zeigt die wesentlichen Bestandteile des TRADE–Traders.

Zwei wichtige, im folgenden näher betrachtete Komponenten des TRADE–Traders sind

das *Typmanagement*, welches ein generelles Typkonzept für DCE–Dienste anbietet, und die *Dienstangebotsverwaltung*, welche eine flexible Verwaltung von Dienstangeboten realisiert und über eine im Vergleich zum Cell Directory Service wesentlich erweiterte Funktionalität verfügt.

4.1 Typmanagement

Der TRADE–Trader besitzt ein dynamisches Typsystem, welches basierend auf dem Diensttypkonzept eine Strukturierung der Diensterbringer erlaubt. Es werden insgesamt drei Schnittstellen angeboten, die nach funktionellen Gesichtspunkten unterschieden sind.

1. Die *Administrationsschnittstelle* stellt Funktionen zum dynamischen Einfügen und Löschen von Dienst– und Schnittstellentypen zur Laufzeit zur Verfügung. Ebenso können *Alias–Namen* verwaltet werden, um eine flexiblere Benennung von Dienst–, Schnittstellen–, Operations– und Attributtypen zu ermöglichen. So kann zum Beispiel für „PrintService“ auch „Druckdienst“ als Diensttypname für Suchanfragen genommen werden. Dies hat den Vorteil, daß die gleichen Typen in verschiedenen Anwendungskontexten benutzt werden können[4]. Die interne Verwaltung der Typinformationen durch den TRADE–Trader erfolgt in *Schnittstellen–* und *Diensttypgraphen*. Diese Graphen spiegeln die Subtypbeziehungen der definierten Schnittstellen– bzw. Diensttypen wider. Die Subtypbeziehungen müssen bei der Definiton neuer Typen *explizit* angegeben werden. Mit dem Diensttyp wird auch gleichzeitig die Beziehung zu dem entsprechenden Schnittstellentyp abgelegt.

2. Die *Browsing–Schnittstelle* dient dem Administrator oder Endbenutzer zum Suchen von Schnittstellen– und Diensttypdefinitionen. So existieren Operationen zum Blättern in den Typgraphen sowie Funktionen zum Suchen von Schnittstellen– bzw. Diensttypen anhand von Operationstypen bzw. Schnittstellentypen und Paaren von Attributnamen und –typen. Auf Basis der gefundenen Typbeschreibungen lassen sich dann diensttypkonforme Dienstnehmer und Diensterbringer entwickeln. Die Diensttypbeschreibungen können auch zur Generierung der RPC–Kommunikationsfunktionen, der sog. *Stubs*, verwendet werden.

3. Die *Trader–Schnittstelle* ist von besonderer Bedeutung für die Vermittlungs– und Typüberprüfungsfunktionen des TRADE–Traders. Diese beinhaltet Funktionen zum Finden *konformer* bzw. kompatibler Diensttypen und zur Überprüfung von Diensttypbeschreibungen auf ihre Korrektheit bzgl. bereits definierter Diensttypen.

Die Speicherung sämtlicher Typmanagementinformationen erfolgt innerhalb des TRADE–Traders als flüchtige Datenstrukturen. Zukünftig soll der in der Abbildung 3 dargestellte Ablagedienst (repository) eine persistente Speicherung der Typinformationen ermöglichen.

4.2 Dienstangebotsverwaltung

Die Dienstangebotsverwaltung beinhaltet Funktionen zur Verwaltung registrierter Dienstangebote, die das Einfügen, Lesen, Löschen und Ändern von Dienstangeboten erlauben.

[4]Dies ist am besten zu vergleichen mit dem „typedef“–Konstrukt in der Programmiersprache C.

Ebenso werden attributbasierte Suchanfragen unterstützt. Neben dem Cell Directory Service dient zusätzlich der X.500–konforme Global Directory Service als Speichermedium zur Ablage von Dienstangeboten, um eine zellübergreifende Dienstverwaltung zu ermöglichen, z. B. in Form einer verteilten Trader–Föderation. Die Dienstangebote werden in *Namenskontexten* in Form von Namenseinträgen gespeichert. Diese Einträge beinhalten unter anderem den angebotenen Dienst– und Schnittstellentyp, die aktuelle Belegung der statischen Dienstattribute sowie die *Schnittstellenreferenz (interface reference)* zum eindeutigen Zugriff auf den Diensterbringer.

Die Programmierung der Dienstangebotsverwaltungsfunktionen wurde weitgehend auf Basis der *XDS– und XOM–Programmierschnittstelle*[5] durchgeführt, die im Vergleich zur NSI–Schnittstelle eine wesentlich flexiblere und mächtigere Programmierung von Funktionen zum Zugriff auf Namensdienste erlaubt. Während die NSI–Schnittstelle speziell für den Zugriff auf den Cell Directory Service zugeschnitten ist, abstrahiert die XDS/XOM–Schnittstelle vom unterliegenden Namensdienst, so daß eine einheitliche Programmierung der Dienstangebotsverwaltungsfunktionen gewährleistet ist. Eine Besonderheit der aktuellen Implementierung ist die Möglichkeit, daß Dienstnehmer auch direkt über die NSI–Schnittstelle auf durch den TRADE–Trader im CDS–Namensraum abgelegte Dienstangebote zugreifen können. Dies hat den Vorteil, daß neue Versionen vorhandener Diensterbringeranwendungen die flexibleren Möglichkeiten der Dienstvermittlung des TRADE–Traders nutzen können, ohne daß bisherige Dienstnehmeranwendungen geändert werden müssen, die weiterhin auf Nutzung des CDS beruhen. Aus diesem Grund werden in den CDS–Namensraum exportierte Dienstangebote mittels der NSI–Schnittstelle abgelegt und anschließend über die XDS/XOM–Schnittstelle um die dazugehörigen Dienstattribute ergänzt.

4.3 Diensterbringerauswahl anhand eines Beispiels

Das folgende Beispiel eines einfachen Druckdienstes verdeutlicht die Funktionalität des TRADE–Traders sowie seine externen Schnittstellen. Für sämtliche Operationen existiert eine Zugriffskontrolle, die auf den Zugriffskontrollmechanismen des DCE Security Service basiert. Dieses ist zum Beispiel für die Administrationsschnittstelle des TRADE–Traders von Wichtigkeit, um unbefugte Manipulationen der Daten zu verhindern. Die Zugriffskontrolle wird jedoch im folgenden nicht näher betrachtet.

Bevor ein Diensterbringer seinen Dienst an den TRADE–Trader exportieren kann, muß vorher der entsprechende Diensttyp bekannt sein. Zuerst muß mit der Operation *insertInterfaceType* der zu dem Diensttyp gehörende Schnittstellentyp „Print“ definiert werden, damit anschließend der Diensttyp „PrintService“ mit der Operation *insertServiceType* eingefügt werden kann. Das Einfügen neuer Schnittstellen– und Diensttypen erfolgt in der jetzigen Implementierung unter expliziter Angabe der Subtypbeziehung zu bereits im TRADE–Trader definierten Typen. Die Übergabe der Typbeschreibungen an die beiden Operationen geschieht über eine spezielle Datenstruktur, die *Dienstrepräsentation*, die sämtliche Typinformationen der Typbeschreibungen beinhaltet. Diese wird auch intern zur Verwaltung der Typgraphen verwendet und dient als universelle Datenstruktur zum Austausch von Typinformationen innerhalb des TRADE–Traders.

Anschließend kann ein Erbringer eines Druckdienstes seinen Dienst beim TRADE–Trader mit der *export*–Operation unter dem Diensttyp „PrintService“ registrieren lassen. Diese erfordert die folgenden Angaben:

[5] X/Open Directory Service und X/Open OSI–Abstract–Data Manipulation

- eine Schnittstellenreferenz, bestehend u. a. aus einer global eindeutigen Objektkennung, einer Rechneradresse und einer Protokollsequenz, zum eindeutigen Zugriff auf die Schnittstelle des Diensterbringers,
- die Diensttypbeschreibung bzw. –repräsentation der Diensttyps „PrintService“,
- ein Dienstangebotsname, z. B. der CDS–Name „/.:/Inf/DBIS/DBISPrintService“, zur Ablage des Dienstangebotes im Namensraum[6], und
- die Belegung der Dienstattribute des Druckdienstes mit aktuellen Werten, z. B. „costPerPage = 0.4“.

Die Angabe der Diensttypbeschreibung dient neben der Angabe des Diensttyps gleichzeitig zur Überprüfung der Korrektheit der übergebenen Dienstattribute. Als Rückgabewert erhält der Diensterbringer die *Dienstangebotskennung (service offer id)*, mit der das Angebot wieder zurückgezogen (*withdraw*–Operation) oder geändert (*replace*–Operation) werden kann.

Ein Dienstnehmer erhält nun mittels der *import*–Operation die für den späteren Zugriff auf die Schnittstellenoperationen eines Druckdiensterbringers notwendigen Bindungsinformationen. Dazu sind folgende Angaben notwendig:

- der Diensttyp des gesuchten Dienstes, hier „PrintService“,
- die Auswahlstrategie, z. B. „random_choice “, nach der geeignete Angebote ausgewählt werden sollen,
- der Suchkontext, der als Ausgangsbasis für eine Suche nach Dienstangeboten über die Dienstangebotsverwaltung im Kontextbaum dient,
- eine Anfrage über die *statischen* und *dynamischen* Dienstattribute, z. B. „costPerPage < 0.5 AND queueLength < 5“ und
- *Optimierungskriterien*, die eine zusätzliche Auswahlqualifikation ermöglichen.

Die Dienstauswahl erfolgt nach dem Prinzip der *gestaffelten Auswertung*, d. h. zuerst werden die zum angegebenen Diensttyp konformen Diensttypen ermittelt. Diese dienen neben den statischen Attributen als Auswahlkriterium bei der anschließenden Suche im Kontextbaum. Je nach Auswahlstrategie werden bei mehreren gefundenen Dienstangeboten zusätzlich die dynamischen Attribute ausgewertet. Diese werden direkt beim Diensterbringer erfragt, um möglichst aktuelle Werte der Attribute zu erhalten. Als letztes werden gegebenenfalls die Optimierungskriterien ausgewertet und das am besten geeignete Dienstangebot ermittelt. Alternativ kann der Diensterbringer aber auch auf die Vermittlungsfunktionen des TRADE–Traders vollständig verzichten, in dem er anstatt eines Suchkontextes den vollen Namenseintrag eines Dienstangebotes angibt. Nach der erfolgten Auswahl werden dem Dienstnehmer die entsprechenden Bindungsinformationen in Form einer Schnittstellenreferenz zum Zugriff auf den Druckdiensterbringer zurückgegeben. Der eigentliche entfernte Prozeduraufruf erfolgt dann wie in Abschnitt 3.1 bereits beschrieben.

[6]Dies ist besonders wichtig für Dienstnehmer, die ohne Nutzung der Vermittlungsmechanismen des TRADE–Traders oder sogar direkt über das CDS auf das Dienstangebot zugreifen wollen.

5 Zusammenfassung und Ausblick

Dieser Artikel hat am Beispiel der Implementierung des TRADE–Traders auf Basis des Distributed Computing Environment gezeigt, wie eine modulare Erweiterung bestehender verteilter Systemarchitekturen um Funktionen des Tradings und eine weitgehende Integration mit den bereits durch die Systemarchitektur bereitgestellten Diensten, insbesondere den vorhandenen Dienstvermittlungs– und Verwaltungsdiensten, erreicht werden kann. Zur Realisierung des TRADE–Traders wurden deshalb so weit wie möglich die durch die DCE–Umgebung bereitgestellten Dienste, insbesondere die beiden Namensdienste, genutzt. Ein erster Prototyp des TRADE–Traders steht zur Zeit zur Verfügung und dient als Grundlage für weitere Forschungs– und Entwicklungsarbeiten im Bereich Dienstvermittlung und –verwaltung. Ein wichtiger Teilschwerpunkt liegt hierbei in der Untersuchung geeigneter Typmodelle und deren Formalisierung. Hierbei werden auch Aspekte der Interoperabilität heterogener Typsysteme in offenen verteilten Systemen betrachtet. Weitere aktuelle Arbeiten beschäftigen sich u. a. mit der Dienstgruppenverwaltung und der Entwicklung effizienter Dienstauswahlstrategien. Ebenfalls wird eine Analyse der Einsatzmöglichkeiten des TRADE–Traders für „Activity Management"–Anwendungen durchgeführt, die spezielle Anforderungen an die Dienstvermittlung und –verwaltung besitzen.

Danksagung
Den Studenten Kai Greese und Stefan Müller sei an dieser Stelle für ihr großes Engagement bei der Prototypentwicklung des TRADE–Traders gedankt.

Literatur

[Ber93] P. A. Bernstein. Middleware – an architecture for distributed system services. Technical report CRL 93/6, Digital Equipment Corporation, Cambridge Research Lab, 1993.

[CW85] L. Cardelli und P. Wegner. On understanding types, data abstraction, and polymorphism. *ACM Computing Surveys*, Band 17, Heft 4, S. 471–522, Dezember 1985.

[Fou92] Open Software Foundation. *Introduction to OSF DCE*. Prentice–Hall, Englewood Cliffs, New Jersey, 1992.

[MJM94] K. Müller, K. Jones und M. Merz. Vermittlung und Verwaltung von Diensten in offenen verteilten Systemen. In *Proceedings GI–Jahrestagung/13th IFIP World Computer Congress*, Informatik Aktuell, Hamburg, Germany, September 1994. Springer–Verlag. (to appear).

[ML93] M. Merz und W. Lamersdorf. Cooperation support for an open service market. In *Proceedings of the IFIP TC6/WG6.1 International Conference on Open Distributed Processing*. North–Holland, Elsevier Science Publishers B.V., 1993.

[MML94] M. Merz, K. Müller und W. Lamersdorf. Service trading and mediation in distributed computing environments. In *Proceedings of the International Conference on Distributed Computing Systems (ICDCS '94)*. IEEE Computer Society Press, 1994.

[Neu89] G. Neufeld. Descriptive names in X.500. *ACM SIGCOM Computer Communication Review*, Band 19, Heft 4, S. 64–71, 1989.

[ODP92] ISO/IEC JTC1/SC21/WG7: Working Document on Topic 9.1 – ODP Trader. International Standardization Organization, November 1992.